PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys

Potter, Douglas ; Stadel, Joachim ; Teyssier, Romain

DOI: https://doi.org/10.1186/s40668-017-0021-1

Potter et al. *Computational Astrophysics and Cosmology* (2017) 4:2
DOI 10.1186/s40668-017-0021-1

Computational Astrophysics and Cosmology
a SpringerOpen Journal

RESEARCH Open Access

PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys

Douglas Potter*, Joachim Stadel and Romain Teyssier

Abstract

We report on the successful completion of a 2 trillion particle cosmological simulation to $z = 0$ run on the Piz Daint supercomputer (CSCS, Switzerland), using 4000+ GPU nodes for a little less than 80 h of wall-clock time or 350,000 node hours. Using multiple benchmarks and performance measurements on the US Oak Ridge National Laboratory Titan supercomputer, we demonstrate that our code PKDGRAV3, delivers, to our knowledge, the fastest time-to-solution for large-scale cosmological *N*-body simulations. This was made possible by using the Fast Multipole Method in conjunction with individual and adaptive particle time steps, both deployed efficiently (and for the first time) on supercomputers with GPU-accelerated nodes. The very low memory footprint of PKDGRAV3 allowed us to run the first ever benchmark with 8 trillion particles on Titan, and to achieve perfect scaling up to 18,000 nodes and a peak performance of 10 Pflops.

Keywords: cosmology; astrophysics; simulations

1 Overview of the problem

The last decade has seen the advent of high precision cosmology, mostly because of the very accurate Cosmic Microwave Background (CMB) experiments WMAP (Spergel et al. 2003) and Planck (Ade et al. 2014). Cosmological parameters, such as the total matter content in the Universe or the Hubble constant are now constrained to within several percent. Although our best fit model, the so-called standard Lambda Cold Dark Matter (LCDM) model, very successfully explains these remarkable observations, it is still based on two mysterious, undetected and elusive components: dark matter and dark energy. The cosmological experiments of the next decade might shed light on this 'dark sector' and possibly revolutionize modern physics. After a decade of CMB experiments, we expect large scale galaxy surveys, such as the ground based Large Synoptic Survey Telescope (LSST Science Collaboration et al. 2009) (LSST), or the two satellite missions Euclid (Laureijs et al. 2011) (in Europe) and WFIRST (Spergel et al. 2013) (in the US), to give new, stronger constraints on our standard cosmological model parameters, possibly below the percent level. Two techniques are considered to measure the clustering of matter as a function of time and scale: weak lensing (WL) and galaxy clustering (GC). Both techniques rely on very accurate theoretical predictions of the non-linear dynamics of the dark matter fluid in an expanding Universe. The more accurate these theoretical predictions are, the more efficient the future large scale surveys will be in solving the mysteries of the dark universe.

Because of the non-linear nature of gravity on these scales, our best theoretical predictions make use of *N*-body simulations: the dark matter fluid is sampled in phase space using as many macro-particles as possible, each one representing a large ensemble of true, microscopic dark matter particles, evolving without collision under the effect of their mutual gravitational attraction. We review in Section 2 the current state of the art in the development of high performance *N*-body codes. Motivated by future dark energy missions, our main goal is to reach an accuracy better than 1% in the power spectrum of the matter density field (see Figure 1) from linear scales (>100 Mpc/h) down

*Correspondence: douglas.potter@uzh.ch
University of Zurich, Zurich, Switzerland

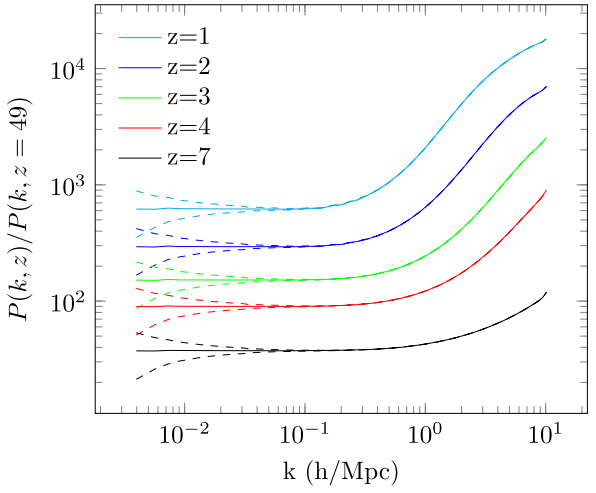

Figure 1 Auto-correlation functions of the density fluctuation in a cosmological box of 3 Gpc sampled with 2 trillion particles simulation at various epoch indicated by the redshift. Dashed lines indicate the statistical errors due to the finite volume of the simulation as well as shot noise. The accuracy of these theoretical predictions is far below the percent level on almost all scales.

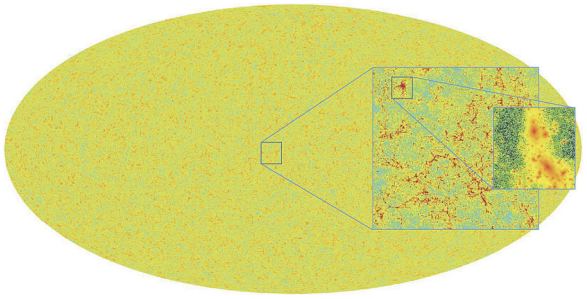

Figure 2 Simulated full-sky matter distribution from a 2 trillion particles simulation. The zoom-in quadrant shows the non-linear, filamentary structure of the universe on small scale.

to strongly non-linear scales ($\simeq$1 Mpc/h). For us to reach these extreme accuracy requirements, we face four different computational challenges: (1) high precision in the gravity calculation, (2) high accuracy in the time stepping, (3) reduce the statistical errors below 1%, which translates to a physical volume of $L \simeq 2$ Gpc/h, and (4) high enough mass resolution, that translates to a large number of particles (for a review see Schneider et al. 2016). The last requirement pushes the limits of what can be achieved on current supercomputers: we need to model accurately dark matter haloes as small as one tenth of the Milky Way mass, which translates into a particle mass smaller than 10^9 $M_\odot$/h, and, for the adopted minimum box size, into a total particle count of $N > 2$ trillion. In the context of future large galaxy surveys, we will need these extreme N-body simulations not just once, but for many different cosmological models, exploring alternative gravity models or galaxy formation scenarios. An additional requirement is a fast enough time-to-solution, so that N-body simulation can optimize and analyze cosmology experiments.

In this paper, we report on the successful evolution of a 2 trillion particles simulation of the LCDM model from $z = 49$ to $z = 0$ *in less than* 80 *h of wall clock time* including on-the-fly analysis, performed on the the Swiss National Supercomputing Center Machine, Piz Daint, using 4000+ GPU-accelerated nodes (see Figure 2). We also report on the first ever benchmark of a 8 trillion particles simulation of the same model, performed on Titan at Oak Ridge using 18,000 GPU-accelerated nodes. Although our 2 trillion particles run represents the minimum requirements for future galaxy surveys, we establish the feasibility of even more extreme particle counts with our 8 trillion particle benchmark. Our tests demonstrate a significant reduction in the time-to-solution and put us in an ideal position to use these extreme N-body simulations for the preparation and the analysis of large galaxy surveys.

2 Current state of the art

N-body simulations in astrophysics have been at the forefront of high performance computing, even before the first digital computer, with the galaxy collision experiment of Holmberg (1941), based on moving light bulbs, and then the heroic 300-particle computer simulation of the Coma cluster performed by Peebles in 1969 (Peebles 1970). Cosmological simulations have been particularly efficient at exploiting the best of each generation of supercomputers, adapting the algorithms to new architectures. In that respect, the number of simulated bodies (or particles) has increased dramatically, owing to the ever increasing performance of supercomputers, but also to the growing efficiency of the N-body solvers. Here, we report the first benchmark ever performed on 8 trillion (8×10^{12}) particles.

In the early 80's, gravity calculations quickly moved away from the accurate but slow $\mathcal{O}(N^2)$ direct interaction (where N stands here for the number of simulated particles) or Particle-Particle (PP) approach, to faster techniques, such as the Particle-Mesh (PM) algorithm (Hockney and Eastwood 1988), based on the Fast Fourier Transform (with $\mathcal{O}(N \ln N)$ efficiency) or the tree code (Barnes and Hut 1986) (also with $\mathcal{O}(N \ln N)$ scaling). Since the PM technique suffers from the limited resolution of the mesh, a hybrid version of PP and PM was later developed, leading to the P^3M technique, which is $\mathcal{O}(N \ln N)$ on large scale and $\mathcal{O}(N^2)$ on small scale (Couchman et al. 1994). The attitude of many generations of code developers since then was to take advantage of the shear performance of the best available computer at that time, but also to reduce drastically the time-to-solution by developing more complex but more efficient algorithms.

In that respect, cosmological simulations are particularly challenging, since they require a fixed simulation time of 13.7 Gyr, namely from the Big Bang until our present epoch. They also require, as explained in Section 1, the largest possible number of particles that can fit in the computer memory. This has led computational cosmologists to develop clever and innovative solutions to optimize the gravity solvers.

Warren and Salmon were among the first cosmologists to be recognized for their parallel tree code's performance, reaching 430 Gflops on ASCI Red (Warren and Salmon 1993, 1997). In 2012, The Millennium XXL simulation (Angulo et al. 2012) was run with 0.3 trillion particles using a specialized version of the GADGET-3 code, based on GADGET-2 (Springel 2005). At about the same time, Ishiyama et al. also achieved 4.5 Pflops with a 1 trillion particle simulation run on the K computer (Ishiyama et al. 2012) for a cosmological simulation using GreeM (Ishiyama et al. 2009), another parallel tree code. Habib et al. (2013) performed a 3.7×10^{12} particle benchmark on a BG/Q system in 2013, this time with a new generation PM+X[a] code called HACC. The HACC code was used in 2014 to produce the Q Continnum Simulation (Heitmann et al. 2014); a full cosmological simulation of 0.55 trillion particles. In 2014 another 1 trillion particle simulation was run by Skillman et al. (2014) using the 2HOT code (Warren 2013). More recently, Bedorf et al. (2014) developed a tree code fully ported on GPUs, and delivered almost 25 Pflops on the Titan supercomputer. These recent achievements demonstrate that tree codes and P^3M codes, both scaling as $\mathcal{O}(N \ln N)$, can deliver significant performance on parallel, and more recently on GPU accelerated, hardware.

In parallel, however, new algorithms have been developed, both for particle and grid-based gravity solver, which in principle could reduce even more the time-to-solution for cosmological simulations. These are the Multigrid (MG) solver (Brandt 1973), which can replace the FFT advantageously, as it scales as $\mathcal{O}(N)$, and the Fast Multipole Method (FMM) (Greengard and Rokhlin 1987; Dehnen 2002) which could deliver the same $\mathcal{O}(N)$ scaling for tree-based codes. While the former, implemented in the Adaptive Mesh Refinement code RAMSES (Teyssier 2001), has been used recently in the 500 billion particles cosmological simulation DEUS (Alimi et al. 2012), the latter, implemented in the PKDGRAV3 code, is the main subject of the present paper.

The $\mathcal{O}(N)$ scaling of FMM clearly offers the opportunity to go to higher particle counts, or to reduce significantly the time-to-solution for a fixed N. Since cosmological simulations are targeting the highest possible value for N, memory is also a strong limitation. The main innovations presented in this paper are (1) a highly performing version of the FMM algorithm, with a measured peak performance of 10 Pflops, and (2) an optimal use of the available memory, allowing us to reach 8 trillion particles on the 18,000 nodes of the Titan supercomputer.

3 Algorithmic improvements

3.1 Fast multipole method

As the 'N' in N-body simulations has increased into the trillions, the asymptotic order of the algorithms to calculate the gravitational forces between the particles is central to having a fast time-to-solution. The $\mathcal{O}(N \ln N)$ gravity calculation of Barnes-Hut (BH) tree-codes, even highly optimized ones which achieve excellent peak performance, are problematic for cosmology simulations. FMM is now vastly superior to the BH for large N, even though it has somewhat lower peak floating point rate than measured by some recent BH codes (Bonsai (Bédorf et al. 2014), 2HOT (Warren 2013)). An aspect of FMM for cosmology simulation is that unlike other codes (BH, P^3M, and tree-PM) the gravity calculation does not take longer as the simulation progresses from the early smooth state of the Universe toward the present day, highly clustered state of matter. This is because FMM *must*, by its scaling with N, be effectively 'blind' to the depth of the tree structure, and hence to the degree of clustering present among the particles in the simulation. FMM and BH are very similar methods; both use particle-particle (PP) interactions for nearby particles and a multipole expansion of the mass within a more distant cell to approximate the force (PC-interactions). However, FMM also considers *cell-cell* (CC) interactions by approximating the potential 'landscape' within a given cell (the sink cell) that is induced by a sufficiently distant multipole (the source cell). While any implementation which uses CC interactions in a sufficiently general way will scale as $\mathcal{O}(N)$ and thus qualifies as an FMM code, several key differences make the FMM as used in PKDGRAV3 highly efficient for very large N simulations.

FMM was originally implemented by Greengard and Rokhlin (1987) using a hierarchy of uniform meshes, but is in fact perfectly suited to implementation using a tree structure as in the BH method. Unlike most tree-codes, PKDGRAV3, uses a binary tree where parent cells are divided along the longest axis into two equal volumed child cells. Using a binary tree as opposed to an oct-tree provides a finer jump in accuracy when going from an expansion based on a parent cell to using the sum of expansions for the child cells. This leads to fewer terms being required to achieve the same force calculation accuracy at the expense of somewhat higher cost in making these decisions (tree walk phase). Another advantage is the simplicity of handling the non-cubical domains that result from *domain decomposition* which divides the simulation volume into sub-volumes which are local to each core. Since we use the traditional ORB (Orthogonal Recursive Bisection) decomposition to balance the number of particles in the do-

mains, this forms the upper part of our global tree structure of which each node and core has a purely local subtree. In fact FMM naturally maximizes locality even within the memory hierarchy as it proceeds down the tree toward the leaf cells since the particles and cells are in a hierarchically sorted order after building the tree. Leaf cells of our tree contain up to b particles (we call this the *bucket size*), where the optimal value is around 16.

Central to the efficiency of a tree code, particularly one using GPU acceleration (see below), is how we create lists of interactions (PP, PC, CC and CP[b]) which when evaluated give us the force on the particles. We walk the tree structure in node-left-right recursive order for sink cells (to which interactions apply) considering source cells that are collected on a checklist. Considering source cells for interactions is traditionally referred to as evaluating an *opening criterion*, but opening a cell (removing it from the checklist and adding its children to the end of the checklist) is only one possible outcome. A source cell on the checklist could also be put onto any of the four interaction lists depending on its distance from the sink cell, or it could remain on the checklist for further consideration by the *children of the sink cell* as we proceed deeper in the tree.[c] Evaluating the opening criterion is a purely arithmetic operation (using AVX/SSE intrinsics for performance and to avoid branches) resulting in a case value of 1 to 6 encoding the outcome for checklist elements. When done this way, these calculations are insignificant to the total computing cost (∼2%). Tree walking begins with the sink cell being the root of the *local* tree of a processor while the checklist contains the *global* root cell of the entire simulation box as well as its 26 (and sometimes 124 depending on accuracy requirements) surrounding periodic replicas.

The actual opening criterion is critical in controlling the distributions of force errors, both in their magnitude and in their spatial correlations.[d] During tree build we calculate a bounding box for each cell and the distance, $b_{\max}$, from the center of mass of the cell (which is always the center of expansions in PKDGRAV3) to the most distant particle in the cell. Based on this we determine an *opening radius* for a cell, $RO = b_{\max}/\theta$, where θ is the traditional opening angle and the force accuracy controlling parameter in the code. If the distance between the source and sink (between centers of mass) are greater than $1.5RO_{\text{sink}} + RO_{\text{source}}$ *and* the bounding boxes are no closer than twice the *softening* (we use 1/50 times the mean inter-particle separation - for a review on the role of softening in N-body simulations see Dehnen and Read 2011), then this is a CC or CP interaction. Note, that there is a deliberate asymmetry here, the factor of 1.5, which controls the spatial correlations in the force errors. For a traditional BH code the force errors typically add up from all directions about a given particle and tend to be correlated spatially with the density of particles. For FMM on the other hand, there is almost no correlation with density (again a working FMM must be blind to tree depth), but we see the tree structure since the expansion of the potential within a sink cell is most accurate at the center of mass and degrades toward the edge of the cell. To reduce this spatial correlation below about 10% of the random errors we have made the acceptance of CC and CP interactions stricter by making sink opening radii larger by this factor. If leaf cells are opened their particles are added to the checklist with $RO_{\text{source}} = 0$ and can later become CP or PP interactions. If a source cell is reached with fewer than g particles (called the *group size*) we proceed no deeper in the tree resolving the remaining checklist into interaction lists, including now PP and PC as well. We have found that a group size of 64, or more generally four times the bucket size, seems to be close to optimal for PKDGRAV3.

Most tree-codes consider multipoles of up to only 2nd order (quadrupoles) which is most efficient for low accuracy force calculation, however for the needed force accuracy of better than 0.1% RMS, going to 4th order moments is more than twice as efficient (Stadel 2001; Dehnen 2002). Not only does the flop/byte ratio increase with order, but also the ratio of FMA (fused multiply add) operations to regular multiply/add, and the number of those compared to the one required $1/\sqrt{|\mathbf{r}|^2}$ increases substantially. The local expansion of the potential about the sink's center of mass is actually done to 5th order, but we do not store this in the tree, since it is sufficient to keep it as a local variable accumulating the CC and CP interactions as we walk the tree. We use single precision in calculating interactions, but all components are accumulated in double precision so we can achieve force errors of around $10^{-5}\%$, well below what is needed for these simulations. To implement periodic boundary conditions, PKDGRAV3 uses a 5th order multipole approximation of the Ewald summation potential (Stadel 2001; Hernquist et al. 1991; Klessen 1997). This requires virtually no data movement and is ideally suited to GPU acceleration, but these calculations must all be done in double precision. Our mixed precision approach serves both to reduce memory usage as well as maximizing the benefit from AVX/SSE as well as GPU floating point hardware.

3.2 Multiple time stepping with dual trees

Cosmological simulations span enormous ranges in density, from very underdense voids, to the centers of dark matter halos that can have densities of 5 orders of magnitude above the mean. This in turn implies that a huge range in dynamical time-scales exist within the simulation. Calculating gravity on all particles at every smallest time-step, while simple from the parallel computing stand-point is very wasteful if the the goal is fast time-to-solution for such simulations. PKDGRAV3 uses individual time-steps per particle, but restricted to being 2^{-l} times a certain *base time-step*, where l is the *rung* to which a particle belongs.

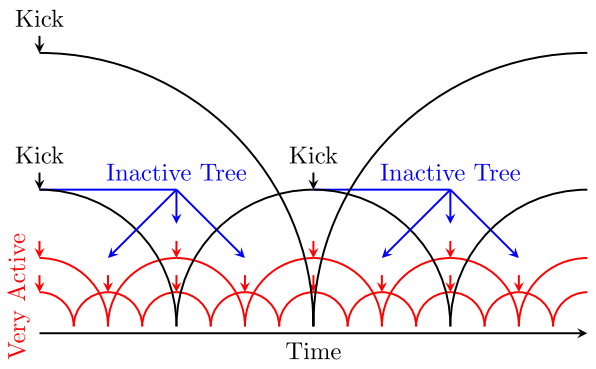

Figure 3 The kick-drift-kick multi-stepping 'umbrella' diagram with the use of dual trees over a single base time-step. Each level of arcs represents one rung and domain decomposition is allowed to move particles between threads *only* at the apex of the black arcs. At these points a single tree is built for all particles in the usual way. Next an inactive (or fixed) tree is built halfway through the black interval and used to calculate force contributions to the remaining red time-steps in a time symmetric way as shown in blue. The red, very active, subtree is all that is built on the shorter very active time-steps where both trees are walked to obtain the combined force.

All simulations presented here use 100 equal base time-steps in proper time to evolve the simulated universes to the present, but many more time-steps are chosen for dynamically active areas of the simulation automatically. We use a hierarchical kick-drift-kick leap-frog scheme shown in Figure 3, where the arrows indicate the force calculations that are applied to advance the velocities. Only the sink cells that contain particles belonging to rung l and higher need to be walked since kicks at higher rungs align in the diagram (we call these the *active* particles). We also need a time-step criterion to decide on which time scale a particle is evolving. The traditional one used in cosmology simulations is based on the particle's softening and the magnitude of its acceleration by $\Delta T_i = 0.2\sqrt{\epsilon/|\mathbf{a}_i|}$. It has been shown that the power spectrum (Schneider et al. 2016) and mass functions of dark matter halos (Reed et al. 2012) converge using this time-stepping criterion. Given the distribution of particles in the rungs of a cosmological simulation, the potential speed-up that is theoretically possible is very large. However, due to the ever greater load imbalance, the decreasing flops/byte and the increase in the relative cost of overheads as the percentage of active particles decreases makes the speed-ups due to multi-stepping less dramatic, but still often a factor of 5× over much of the simulation. We discuss a novel method of reducing the most significant overhead, namely the tree build time, by building a second smaller tree only for very active particles.

With any multi-stepping code, there will be rungs with very little gravity work to do since only a small percentage of the particles are active. Nevertheless, the tree must still be built, walked, and the forces evaluated. The time needed for the force evaluations reaches a trivial stage while building a full tree still takes the same amount of time. As the number of tree builds scales as 2^l, the tree build cost quickly starts to dominate. We build a single second very active tree when the number of particles on a rung drop below a certain threshold (5% seems to be a good value).[e] The inactive particles are drifted half-way along their trajectory and a fixed tree built as shown in Figure 3. Subsequently, only an active tree is built until it is time to kick the fixed particles at which point they are drifted through the remaining half of their trajectory. It is very important to construct the second tree by traversing the fixed tree and using the same geometric structure. This assures that cells in the very active tree are approximately the same size as cells in the fixed tree in a given region of space (somewhat similar to the construction of *graded* trees in AMR codes). Not doing this sometimes results in an unreasonably high number of interacting particles.

3.3 GPU acceleration

While other codes (Bédorf et al. 2012) have attempted to use the GPU for tree related operations, we made the deliberate decision to split the work between the CPU and GPU in a manner that compliments their strengths. Walking a tree is geometrically complex, exhibits branch divergence, and requires accessing tree nodes on remote processors. Conversely, evaluating interactions and multipoles is ideal work for the GPU. The GPU work consists of PP interactions, PC interaction and the periodic boundary condition evaluation (Ewald). PKDGRAV3 monitors the flop/byte ratio of interaction lists as they are generated and in the rare case that this falls below an optimal threshold then the work is instead issued directly to the CPU. This allows the GPU to concentrate on work packages that can keep utilization high resulting in a lower overall run-time. The operations are fully asynchronous allowing almost perfect overlap of compute and communication with the GPU.

3.4 Memory

With the use of FMM, multiple time-steps and GPU acceleration the major limiting factor for these simulations is the amount of available memory on each node. PKDGRAV3 has been developed to minimize memory usage per particle (see below) and allow the maximal use of the available memory for particles. This includes: (1) bypassing Linux file I/O and instead using direct I/O to have complete control of file buffering, (2) making memory balancing the primary goal of domain decomposition, (3) reducing the memory usage by the tree, (4) partitioning memory very carefully on a node and in most cases preallocating it. Careful consideration is also given to the memory usage of the many analysis tasks that are performed during the run including group finding, light cone generation as well as the storage required to generate the initial

condition at the beginning of the simulation.[f] The particle light-cone for example uses two relatively small buffers and asynchronous I/O where particles are added to one of the buffers while the other buffer is being written in the background. In this way, I/O and computing can seamlessly overlap. This is possible because particles in the light-cone are output exactly when the light surface intersects them, rather than at fixed points during the simulation. This has the added benefit of outputing particle at the exact correct time, rather than in slices of fixed time as is often done. Minimizing the memory use per particle has the nice side benefit of increasing performance in the tree building and tree walking phases of the code that are strongly affected by the efficiency of transferring to and from memory.

Storage for particles is divided into two regions; a 'persistent' area containing properties that must persist between steps, and 'ephemeral' storage used for certain algorithms, for example group finding, where the intermediate data can be forgotten when the calculation ends. In the persistent storage, we identified *position*, *velocity*, *group id*, and current *rung*. Velocities can be stored as single-precision float values without affecting the results. Positions are trickier. It is necessary to resolve well below the softening scale which in our case is one part in a million.[g] We would like to achieve a resolution of perhaps a hundredth of the softening length which would require of order 27 bits of precision, greater than that provided by single precision. We convert double precision float values between integer coordinates which provides 32 bits[h] of precision which is more than sufficient. We have checked that this simple particle compression scheme does not affect their trajectories in any significant way for cosmological N-body simulations. The ephemeral storage can vary between zero bytes (when no analysis is required), to 4 bytes if power spectra or group finding is needed up to 8 bytes for other algorithms. Future analysis may require more memory in which case the ephemeral area would increase. As a special case, it is possible to use part of the tree memory for algorithms when a tree is not required (when generating initial conditions for example). We also need a small amount of memory for explicit communication buffers as well as room for the tree (which tends to grow as structure forms). All told, the simulation can be run with approximately 62 bytes per particle as summarized in Table 1. A simulation of 2 trillion particles can be easily run on Piz Daint (which has 169 TB of memory) while an 8 trillion particle simulation can be run on Titan (which has 584 TB).

Table 1 Memory requirements per particle

Persistent 28 bytes	Ephemeral 0-8 bytes	Tree 25 bytes	Buffers ∼5 bytes

Buffers are $\mathcal{O}$(125 MB) per thread. Here we assume 16 threads with **5×10^8** particles on a 32 GB node.

4 Performance results

At the time this paper was written, Titan (Oak Ridge National Labs, USA) was the second fastest supercomputer in the world with a measured LINPACK performance of 17.59 Pflops and was used for most of the performance benchmarks reported here. It is a Cray XE7 system with 18,688 compute nodes and a Gemini 3-D Torus network. Piz Daint (Swiss National Supercomputing Center), a Cray XC30 with 5,272 compute nodes connected via the Aries Dragonfly (multilevel all-to-all) network is currently the 7th fastest computer in the world and is being used for the 2×10^{12} particle production run, upon which the benchmarks are based (the same mass resolution). The 282 node Cray XE6, Tödi (Swiss National Supercomputing Center), is useful for development and testing of large scale applications for Titan, being a much smaller instance of this system. The individual nodes of these three machines are similar, each having 32 GB of main memory a single CPU as well as an nVidia K20X GPU accelerator. Titan and Tödi use the AMD Opteron models 6274 and 6272 with a clock speed of 2.2 and 2.1 GHz respectively while Piz Daint uses an Intel Xeon E5-2670 with a variable clock speed ranging from 2.6 GHz up to 3.3 GHz (3.0 GHz with all cores active). Titan has the largest total system memory of 584 TB which allows for a production simulation with PKDGRAV3 of 8×10^{12} particles with a time-to-solution of 67 hours. The detailed benchmark and scaling results presented below will establish that such a high resolution simulation is indeed possible within this projected time.

All of these machines have multiple CPU cores on each node, and the trend is for this number to increase. PKDGRAV3 employs a 'hybrid' pthreads/MPI model with a single MPI thread per node, and threads on the same node exchange data using shared memory. The work is still divided between the individual threads in the same way it would have been with an MPI-only version which results is the same (perfect) scaling. While the dedicated MPI thread is only 25% utilized, not allowing it to participate in the gravity calculation has the effect of dramatically reducing message latency and increases overall performance.

4.1 Timing measurements

In the following sections, timing information is collected through the use of timers in the code. The run-time is divided into four phases - load balancing, tree construction, force evaluations, and analysis. The first three phases are carefully timed and included in these results. The fourth, analysis, is not included as it can vary significantly depending on which analysis needs to be performed. If more sophisticated analysis 'instruments' (by which we mean further software to perform on-the-fly analysis) were to be attached to PKDGRAV3 then the time would increase from the roughly 25% for our current production simulations

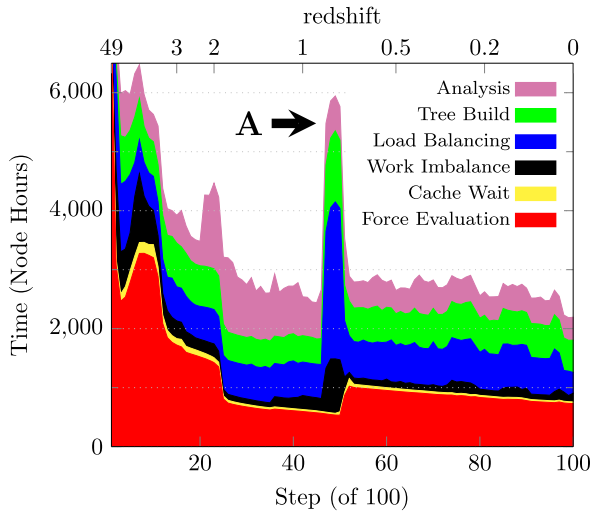

Figure 4 Distribution of run-time between various phases of the calculation. The red, yellow and black regions are force calculation, the blue region is for balancing the work, the green region is tree build and the magenta region is on-the-fly analysis. The feature indicated by **A** is described in Appendix 1.

where halos and halo properties were identified using a friends-of-friends group finder, the power spectrum was measured, and the particle light cone and healpix maps were output.

We also use the high-resolution on-chip timers to measure sub-phases, in particular we are able to distinguish how much time is spend calculating forces, how much time is spent waiting for communication requests to complete, and how much time is wasted at the end of a step because of load imbalance. We discuss the later two only cursorily as they have a nearly insignificant effect on time-to-solution as shown in Figure 4. The timings for analysis include the necessary I/O; indeed this can easily be seen in the figure where the analysis time suddenly increases as the 'particle light-cone' begins. Raw particle output is written to disk only when checkpointing which takes takes 30 minutes per checkpoint for the two-trillion particle simulation run on Piz Daint. This accounts for a roughly 5% cost increase depending on how frequently checkpoints are written. Initial conditions are also generated by PKDGRAV3 in memory at the start of the simulation, a procedure which takes approximately 5 minutes.

4.2 Simulation accuracy

While it is possible to speed-up the simulations by relaxing the accuracy requirements, taking either fewer time-steps or increasing θ, thereby reducing the force accuracy, we emphasize here that we do not do this in any of the benchmarks. We run all benchmarks with the *same* run parameters that we are using for our 2×10^{12} particle production simulation which will serve as the first reference simulation for the Euclid mission. These run parameters were carefully tuned in a series of lower resolution simulations (Schneider et al. 2016) where three different codes were run and produced a result where the power spectrum agreed to better than one percent. At very early times ($z > 20$), when the Universe is very homogeneous, the forces from opposing directions very nearly cancel and a tree code must use a stricter opening criterion in order to attain the same accuracy in the force. Additionally, small errors in the initial non-linear growth of these first structures amplify during the further evolution and can lead to errors greater than 1% in the power spectrum by the end of the simulation if the force accuracy and time-stepping is not conservative enough. We set $\theta = 0.40$ for $z > 20$ (to 1% age of the Universe), $\theta = 0.55$ for $20 > z > 2$ (to about 20% age of the Universe), and $\theta = 0.70$ for the remaining 80% of the evolution. We note that these quoted θ values apply for the 5th order expansion used in PKDGRAV3 and result in much more accurate forces than in the traditional quadrupole based BH codes. These transitions in the force accuracy and cost per step can clearly be seen in Figure 4.

The particle mass remained fixed at 10^9 solar masses for all benchmarks as previously mentioned. This is small enough to converge on the power spectrum to 1% and to resolve objects down to the needed scale to produce so called mock galaxy catalogues (Fosalba et al. 2013) for Euclid, weak lensing maps and statistics for galaxy clusters. It should be strongly emphasized that the smaller the mass scale that is simulated, the *harder* the simulation becomes, or comparing simulations of the same N, the one with the *smaller* box size is the more challenging. While PKDGRAV3 is independent of the degree of clustering in the force calculation, the peak densities within a simulation of smaller particle mass are higher and therefore the number of time-steps needed increases. We find that for PKDGRAV3 decreasing the box size by a factor of two while keeping the same number of particles results in an approximately 50% longer runtime.

In Figure 4 we show the actual time spent in different tasks integrated over each base time-step for our completed 2×10^{12} particle production run on Piz Daint. Force calculation (in red) dominated the early time-steps, while later there is a near equal balance between it, tree building (green), domain decomposition (blue) and all of the on-the-fly analysis (magenta). The yellow/black contribution shows time spent waiting either because the work is not completely balanced (black), or because of communication delays (yellow).

It used to be the case that analysis was performed by post-processing the results, but with the ever increasing simulation sizes writing raw simulation output to the disk is no longer feasible, since this would vastly dominate the time-to-solution. The spike in the magenta analysis

time at around step 20, for example, is a result of particle 'light cone' analysis kicking in. Our friends-of-friends group finder, and the analysis on the resulting dark matter halos that are found by it, were also completely rewritten to be competitive with the other tasks (otherwise it would have been the dominating task at this scale). It is interesting to see that such analysis tasks must not be neglected when considered fast time-to-solution, since even when highly optimized, they contribute significantly to the total run time.

While tree building and domain decomposition times remain reasonably constant, gravity calculation changes for two reasons. As mentioned previously the force accuracy requirement changes (most notably at around step 24) when much of the mass is in viralized dark matter halos. The second reason is that the time-step also scales with the mean density of the Universe ($\Delta T \propto 1/\sqrt{\rho}$) which is decreasing very rapidly early on. This means that at the beginning of the simulation there are a lot of particles at very small time-step rungs which results in a heftier gravity calculation contribution. This never stops so the time per step will continue to decrease by a modest amount until the very end. We note again, that this is quite in contrast to what is observed for BH and P^3M codes. The onset of structure formation, which goes in the other direction to increase the number of time-steps, can be seen between steps 5 and 10 when the gravity time increases even though there has been no change in the force accuracy during this time. Structure formation stabilizes, in the sense that all density peaks have been established and most of the mass that can end up in dark matter halos is bound up in them.[i] Finally, the modest cost of tree building seen here is only possible when using the dual tree method described previously. Without this innovation the tree build contribution would be 3 times larger.

4.3 Multi-stepping and dual tree boost

Although there were 100 base steps, PKDGRAV3 uses a multi-stepping scheme where particles choose their own time-step rung based on the time-step criterion discussed previously. For the benchmark simulations this results in effectively $5{,}000 \pm 10\%$ time steps. For rungs with very few particles, each step can take a fraction of a second. While the time for a full gravitational calculation can be in the range of minutes, the average time per step is of order 50 seconds, including tree build and domain decomposition (but not including on-the-fly analysis). For simulations of this type, multi-stepping results in an effective speed-up of between 4x and 5x when compared to taking single time-steps.

As discussed earlier, the tree building phase can begin to dominate when multi-stepping. A complete gravity step takes of order two minutes, while constructing the tree takes more like 25 seconds. When multi-stepping, some of the gravity calculation takes less than a second while the tree building time does not vary. By constructing a second tree for the very active particles, the tree build time is reduced to one second for these critical sub-steps. The method results in an additional 26% decrease in the overall time-to-solution.

4.4 GPU boost

PKDGRAV3 is already highly optimized for SIMD type instructions, such as SSE and AVX, and because of mixed-precision (float/double) code, the performance boost is already a factor of eight for some parts of the calculations. Because not all calculation are FLOP dominated, for example load balancing and tree construction, the effective speed-up is more like $3\times$. By using the GPU, the situation is dramatically improved. For the Tödi simulation shown in Figure 7, a single force evaluation[j] that took 1,138 seconds using only the CPU, takes 119.5 seconds when using the GPU - a speed-up of $9.5\times$. A complete step, including all phases (gravity, tree construction and load balancing), takes 1,629 seconds with the GPU compared to 6507 with the CPU only, resulting in a $4.0\times$ improvement in the time-to-solution.

Part of the GPU work scheduling involves shunting work to the CPU when appropriate. If the number of particles is too small (1 or 2), then the CPU will do the work. If the GPU is too busy, detected when too many work packages are scheduled on the GPU but not yet complete, then pieces of the interaction list that do not evenly align with a WARP[k] are done by the CPU instead. While it is possible to push more work to the GPU, and thus increasing the total FLOP rate, this comes at the expense of an increased time-to-solution.

4.5 Scaling

To perform the very largest simulations, it must be demonstrated that PKDGRAV3 can efficiently scale up to the task. Weak scaling was measured by starting with a $1{,}000^3$ simulation (10^9 particles) and running it on two nodes to measure the gravity calculation times. The simulation was then scaled upward by scaling the total number of particles and the total number of nodes by the same factor. The simulations run are outlined in Table 2. Here we see that the total run time remains constant as the simulation size is increased, which is expected for an $\mathcal{O}(N)$ method which has low parallel overheads and good load balance. We include a direct comparison with the HACC (Habib et al. 2013; Habib et al. 2014) and 2HOT (Skillman et al. 2014) codes. The weak scaling runs for PKDGRAV3 were all performed with 4.7×10^8 particles per node, the HACC benchmarks with 0.32×10^8 particles per node, and the 2HOT simulation with 0.81×10^8 particles per node. As the weak scaling of these codes is essentially perfect, the total run-time does not change when using the same number of particles

Table 2 Weak scaling performance on Titan with 4.7×10^8 particles per node

Nodes	N_p	Mpc	Time (secs)	Science Rate[1]
2	1.0×10^9	250	124.9	4.00×10^6
17	8.0×10^9	500	117.4	4.02×10^6
136	6.4×10^{10}	1,000	117.9	3.98×10^6
266	1.3×10^{11}	1,250	125.1	3.76×10^6
2,125	1.0×10^{12}	2,500	124.0	3.79×10^6
7,172	3.4×10^{12}	3,750	123.2	3.82×10^6
11,390	5.4×10^{12}	4,375	126.6	3.72×10^6
18,000	8.0×10^{12}	5,000	120.1	3.70×10^6

The science rate remains constant.
[1]In particles per second per node.

per node. This is the most relevant scaling for these types of cosmological simulations as it is typical to be memory limited due to the desire for high resolution as well as large volume. For the same simulation size, 1.0×10^{12} particles, the results from HACC, 2HOT and PKDGRAV3 are similar with a science rate (millions of particles per second per node) of 1.7 for HACC,[l] 1.2 for 2HOT,[m] and 3.8 for PKDGRAV3. As the HACC and 2HOT benchmarks are not particularly current we would expect that today improved results could be presented by these authors. When the total number of particles per node was kept fixed at 4.7×10^8 as was the case for the weak scaling tests, an entire simulation would run to completion in 67 hours *regardless of size*.

To measure strong scaling, we start with a series of simulations with $1{,}000^3$, $2{,}000^3$ and $3{,}000^3$ particles (10^9, 8×10^9 and 2.7×10^{10}) and run them on the smallest number nodes where they will fit (so 4.7×10^8 particles per node). The number of nodes is then incrementally increased. As shown in Figure 5, PKDGRAV3 shows excellent strong scaling up to a factor of several hundred. This allows us to reduce the wall-clock time of simulations by up to a factor of a hundred or more by simply increasing the number of nodes. Recall that when using the most particles possible per node and hence the maximum wall clock time, a simulation will take approximately 67 hours. Using 10 times as many nodes results in only a 25% penalty meaning a simulation would take less than 10 hours. Using 100 times as many nodes carries a 70% penalty, meaning a simulation would take slightly longer than an hour.

4.6 Raw performance

With PKDGRAV3, a great deal of effort has gone into algorithmic improvements to try to avoid, wherever possible, doing unnecessary work. This has the effect of greatly complicating the data structures making it more difficult to achieve high raw flop counts. Nevertheless, for a code to achieve high performance, the raw performance must be at least competitive.

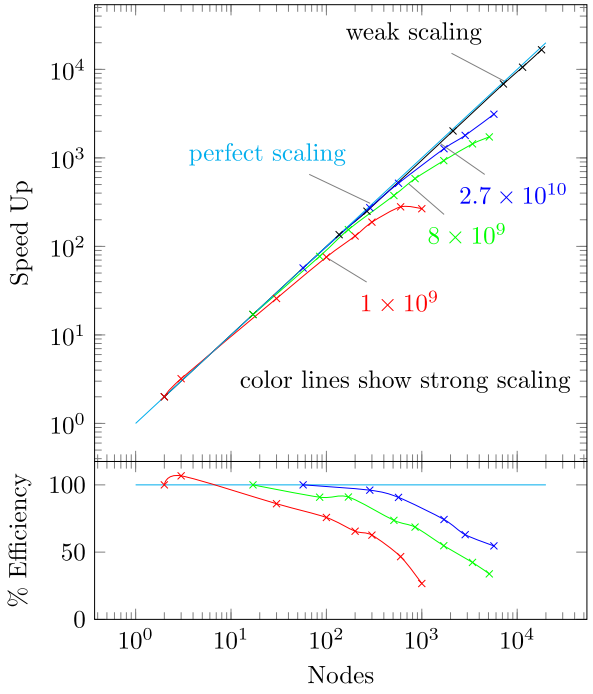

Figure 5 Weak and strong scaling. Perfect $\mathcal{O}(N)$ scaling should follow the slope of the 'perfect scaling' line. PKDGRAV3 exhibits perfect weak scaling and excellent strong scaling out to 300 times the number of nodes. This translates into node memory usage starting at 30 GB and scaling to 0.1 GB.

Table 3 Flop counts by phase

Phase	+ − ×	√	÷	FLOPs
Particle/Particle	46	1		53
Particle/Cell	208	1		215
Cell/Particle	206	1		213
Cell/Cell	472	1		479
Ewald iteration	433	1	2	510
Opening criteria	97			97

Bold entries use *double* precision calculations.

To determine the number of floating point operations used, the AVX version of the code was examined to determine how many floating point instructions were required for each phase of the calculations. Most operations, including addition, subtraction and multiplication count as a single flop. The reciprocal square root is scored as seven flops while a division is scored as 35 flops. The totals for each phase are shown in Table 3. In addition, floating point operations were divided into single and double precision, and totaled separately for the CPU and GPU.

In Table 4 we show the peak performance achieved for various simulation sizes where the number of particles is optimized to fill a node. We also show the wall-clock time required to calculate the forces for a single particle.

Table 4 Performance on Titan

Nodes	N_p	Mpc	TFlops	Time/Particle
2	1.0×10^9	250	1.2	125 ns
17	8.0×10^9	500	10.3	14.7 ns
136	6.4×10^{10}	1,000	82.2	1.84 ns
266	1.3×10^{11}	1,250	152.5	1.00 ns
2,125	1.0×10^{12}	2,500	1,230.3	0.124 ns
7,172	3.4×10^{12}	3,750	4,130.9	0.0365 ns
11,390	5.4×10^{12}	4,375	6,339.2	0.0236 ns
18,000	8.0×10^{12}	5,000	10,096.2	0.0150 ns

Total measured TFlops as well as the wall-clock time to calculate the forces for a single particle.

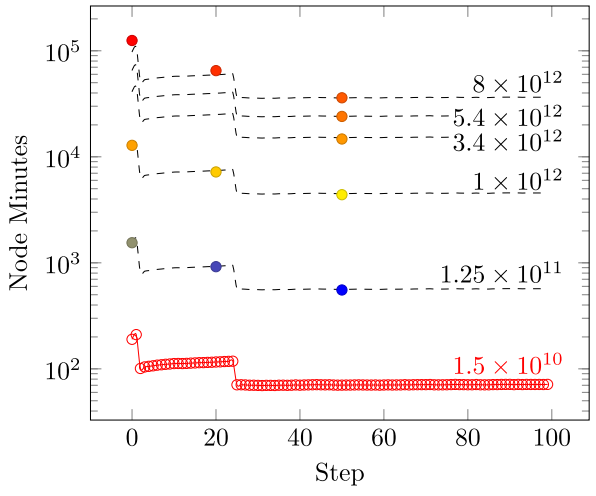

Figure 6 Gravity time per step. Circles are measurements while dashed lines are predictions.

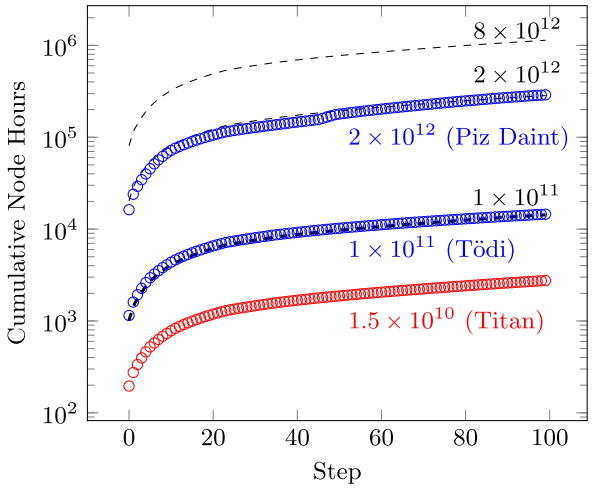

Figure 7 Total runtime in node hours (wall clock × number of nodes). The red circles are measurements of a simulation run on Titan with 320 nodes. The dashed lines are predictions based on both weak and strong scaling. The much larger Tödi simulation was run on 214 nodes while the even larger Daint run used 4,900, and later 4,000 nodes. All measurements show excellent agreement with the predictions.

While PKDGRAV3 does use mixed precision float code, the measured 10 Pflops compares quite well with the 17.59 measured LINPACK performance. The exact blend of single and double precision varies depending on the required accuracy as discussed earlier, for a typical step more than 90% of the flops are calculated on the GPU, and almost 40% of those are in double precision.

4.7 Time to solution

To measure time-to-solution, we start by running a complete simulation at a lower resolution. Because of the physical processes involved, the timings for each step can be roughly broken into three distinct phases corresponding to different integration accuracy domains. In Figure 6 we show the timings for gravity calculations in total node hours during each of the 100 main steps. As PKDGRAV3 is an $\mathcal{O}(N)$ code, these timings are then scaled linearly by the problem size to estimate how long the force calculations will take. The estimates are verified by running the force calculation at sampled points, and comparing them to the estimates.

This can be seen in Figure 6. The hollow circles represent the measured timing for a force calculation on all particles throughout a simulation of $2{,}500^3 (1.5 \times 10^{10})$ particles. As is clearly apparent in the figure, the time required perform the gravity calculation is extremely stable. The three different 'steps' correspond to the accuracy requirements (high redshift requires increased accuracy). The timings are given in node minutes (wall clock time multiplied by the number of nodes).

The dashed lines show predictions for the force evaluations at increasing resolutions made by scaling the low resolution simulation by the problem size. Measurements were then taken a several points at each resolution shown by the solid circles. The prediction and measurements agree perfectly.

In Figure 7, the cumulative node hours for the reference simulation is plotted. In order to complete the simulation quickly, it was run on 320 nodes, even though it could have fit in as few at 32. We make predictions for how long simulations of various sizes, namely 10^{11}, 2×10^{12} and 8×10^{12} would take based on the weak scaling. As this is now in the strong scaling regime we further correct the prediction by assuming that it could be run 24% faster (recall that the penalty for strong scaling by a factor of 10 is 24%). These predictions are shown as dashed black lines.

A complete simulation of 10^{11} particles was run on Tödi using 214 nodes which corresponds to full memory usage of 4.7×10^8 particles per node. The measured performance of the simulation shows perfect agreement with the

estimate. A further simulation was run on Piz Daint using 2×10^{12} (2 trillion) particles. Due to the slight differences in architecture, the simulation actually beats the prediction by a modest amount. We expect this is due to the slightly better AVX performance on the CPU, and perhaps to a lesser degree the network.

The end result is that we have high confidence that an 8 trillion particle simulation is possible on Titan using 18,600 nodes, and it will take of order 67 hours with some additional time for on-the-fly analysis which would vary depending on the exact analysis done. This also means that a 1 trillion particle simulation run on Titan using all nodes could be completed in under 10 hours.

5 Implications

In order to achieve the results presented here, significant *refactoring* of the code was required. Tracking the progress in N-body simulations over time, a performance doubling time of roughly 1 year is observed. This rate, which exceeds Moore's Law can only continue if further efforts are made to refactor algorithms for new computing hardware. These gains can also be pushed forward by *co-design*, where computing hardware and algorithmic developments are considered as a single design process.

The new time-to-solution of these simulations is a game changer as far as the way theory is used in cosmological measurements. For the first time simulations will not only be used to help understand effects or to make some predictions, but will be needed to extract fundamental physical parameters from future survey data. They must become part of the data analysis pipelines.

Another implication for the future is that time-to-solution will continue to decrease as greater computational speed will out-strip any possible increase in memory size. Our memory footprint is about as low as it is possible to go per particle, so that the time-to-solution for these simulation can only decrease from this point on. We expect to run such simulations within 8 hours or less within the decade. This also means that raw data will never be stored and post-processed. Instead data analysis 'instruments' will be attached to the code and the simulations will be rerun, perhaps several times with different 'instrumentation'. This is starting to happen and is a true paradigm shift in the field of simulations.

Appendix 1: Computational challenges

During Grand Challenge simulations such as this one, there are inevitably problems encountered, and such was the case here. In Figure 4, the time per step suddenly increases at step 46 as indicated by the arrow labelled **A**. This was caused by one of the nodes performing in a substandard way which resulted in the entire simulation to take twice as long, as the other nodes were waiting for this node to complete its share of the work. The exact cause of this problem is not known, and will never be known, but it was very likely a rogue process that was left running on the node that stole processing cycles. This problem disappeared when the simulation was restarted without this node.

The second problem occurred shortly thereafter, around step 50, and was a result of the increase in efficiency as the simulation progressed. In Figure 4 we see that the gravity calculation time drops dramatically between step 0 and step 20 as structure forms and the effect of the initial condition grid is no longer relevant allowing the force accuracy to be relaxed. At some point, the amount of work being shipped to the GPU reaches a threshold that triggers a not yet understood problem with the GPU device. When this threshold is reached, the GPU will, very rarely, accept work but never complete it. By sending work in a more controlled fashion, this problem is eliminated or vastly reduced allowing the simulation to run to solution, but with slightly decreased performance. The cause of this is still under investigation.

Although these two problems seem dramatic, they had very little impact on the total run-time as can be seen in Figure 7. The simulation was on track to slightly beat the estimate, but the two problems conspired to slightly increase the total run-time causing it to take almost exactly the amount of time predicted.

Appendix 2: Simulation cost

The measurement of the total cost of a simulation has become more complicated with the introduction of hybrid computing. The historical method used was to take the total number of CPUs (and later CPU cores) and multiply this by the wallclock time. Thus, a simulation that runs with twice as many cores in half the time has the same computational cost. This also addresses the situation where the number of cores changes between different allocation periods.

Hybrid nodes containing both a CPU with multiple cores, and a GPU device make this simple calculation more difficult. While some supercomputer centers simply assign a core count to the CPU, and a different core count to the GPU, there are others that avoid the problem altogether by quoting 'node hours' instead; a unit that is understood to mean the CPU and GPU resources of a single node. We have adopted this nomenclature, as it is non-ambiguous, and it allows the reader to easily make whatever transformation they so desire.

Acknowledgements

We are indebted to the folks at the Swiss National Supercomputer Center, specifically Thomas Schulthess, Maria Grazia Giuffreda and Claudio Gheller for the support and advice, and for resurrecting Tödi. We would also like to thank Jack Wells for arranging access to Titan.

Funding
Parts of this work were supported by a grant from the Swiss National Supercomputing Center (CSCS) under project ID S592. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Availability of data and materials
The code used to run these simulations (PKDGRAV3) is available on bitbucket (www.pkdgrav.org).

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
DP and JS jointly developed the code PKDGRAV3. DP ran the production simulations and benchmarks. RT, JS and DP contributed to the text of this paper. All authors read and approved the final manuscript.

Endnotes
[a] HACC can use a number of hybrid PM methods including P^3M or TreePM with or without GPU or other accelerators.
[b] Cell-particle interactions are the mirror image of particle-cell interactions; they are the expansions of the potential within the sink cell induced by a single source particle.
[c] It is rare for a cell to stay on the checklist for more than a few levels as it will end up on one of the interaction lists or be opened.
[d] Ideally we want spatially uncorrelated errors, but this is as impossible to attain as is having all force errors precisely at the desired truncation error.
[e] The dual trees are only constructed if there are at least two rungs below the fixed rung, otherwise there is no performance benefit.
[f] PKDGRAV3 uses the 2LPT method which requires 13 FFT operations and with some juggling can be done with 36 bytes per particle.
[g] Grid size of 1/20,000 × softening scale of 1/50.
[h] Actually slightly less as the representable box must be slightly larger than the simulation volume.
[i] Larger and larger structures continue to form but this does not affect the time-step hierarchy.
[j] At late time when gravity calculations no longer dominate the run-time; speed-up at earlier times is higher.
[k] If the interaction list has 655 elements for example, then 640 would be calculated by the GPU, and 15 by the CPU.
[l] Private communication.
[m] Table 1 of Skillman et al. (2014).

Publisher's Note

Received: 15 September 2016 Accepted: 27 April 2017
Published online: 18 May 2017

References
Ade, PAR, et al.: Planck 2013 results. XVI. Cosmological parameters. Astron. Astrophys. **571**, 16 (2014). doi:10.1051/0004-6361/201321591; arXiv:1303.5076
Alimi, JM, Bouillot, V, Rasera, Y, Reverdy, V, Corasaniti, PS, Balmes, I, Requena, S, Delaruelle, X, Richet, JN: DEUS full observable LCDM universe simulation: the numerical challenge (2012). arXiv:1206.2838
Angulo, RE, Springel, V, White, SDM, Jenkins, A, Baugh, CM, Frenk, CS: Scaling relations for galaxy clusters in the Millennium-XXL simulation. Mon. Not. R. Astron. Soc. **426**(3), 2046-2062 (2012). doi:10.1111/j.1365-2966.2012.21830.x; arXiv:1203.3216
Barnes, J, Hut, P: A hierarchical O(N log N) force-calculation algorithm. Nature **324**(6096), 446-449 (1986). doi:10.1038/324446a0
Bédorf, J, Gaburov, E, Portegies Zwart, S: Bonsai: a GPU tree-code. In: Capuzzo-Dolcetta, R, Limongi, M, Tornambè, A (eds.) Advances in Computational Astrophysics: Methods, Tools, and Outcome. Astronomical Society of the Pacific Conference Series, vol. 453, p. 325 (2012). arXiv:1204.2280
Bédorf, J, Gaburov, E, Fujii, MS, Nitadori, K, Ishiyama, T, Portegies Zwart, S: 24.77 Pflops on a gravitational tree-code to simulate the Milky Way Galaxy with 18600 GPUs In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 54-65 (2014). doi:10.1109/SC.2014.10; arXiv:1412.0659
Brandt, A: Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In: Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, vol. 19, pp. 82-89. Springer, Berlin (1973). doi:10.1007/BFb0118663
Couchman, HMP, Thomas, PA, Pearce, FR: Hydra: an adaptive-mesh implementation of PPPM-SPH. Astrophys. J. **452**, 797 (1994). doi:10.1086/176348; arXiv:astro-ph/9409058
Dehnen, W: A hierarchical $\mathcal{O}(N)$ force calculation algorithm. J. Comput. Phys. **179**(1), 27-42 (2002). doi:10.1006/jcph.2002.7026; arXiv:astro-ph/0202512
Dehnen, W, Read, JI: *N*-Body simulations of gravitational dynamics. Eur. Phys. J. Plus **126**(5), 55 (2011). doi:10.1140/epjp/i2011-11055-3 arXiv:1105.1082
Fosalba, P, Gaztanaga, E, Castander, FJ, Crocce, M: The MICE Grand Challenge Lightcone Simulation III: galaxy lensing mocks from all-sky lensing maps. Mon. Not. R. Astron. Soc. **447**(2), 1319-1332 (2013). doi:10.1093/mnras/stu2464; arXiv:1312.2947
Greengard, L, Rokhlin, V: A fast algorithm for particle simulations. J. Comput. Phys. **73**(2), 325-348 (1987). doi:10.1016/0021-9991(87)90140-9
Habib, S, Morozov, V, Frontiere, N, Finkel, H, Pope, A, Heitmann, K: HACC. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13, pp. 1-10. ACM, New (2013). doi:10.1145/2503210.2504566
Habib, S, Pope, A, Finkel, H, Frontiere, N, Heitmann, K, Daniel, D, Fasel, P, Morozov, V, Zagaris, G, Peterka, T, Vishwanath, V, Lukic, Z, Sehrish, S, Liao, WK: HACC: simulating sky surveys on state-of-the-art supercomputing architectures. New Astron. **42**, 49-65 (2014). doi:10.1016/j.newast.2015.06.003; arXiv:1410.2805
Heitmann, K, Frontiere, N, Sewell, C, Habib, S, Pope, A, Finkel, H, Rizzi, S, Insley, J, Bhattacharya, S: The Q continuum simulation: harnessing the power of GPU accelerated supercomputers. Astrophys. J. Suppl. Ser. **219**(2), 34 (2014). doi:10.1088/0067-0049/219/2/34; arXiv:1411.3396
Hernquist, L, Bouchet, FR, Suto, Y: Application of the Ewald method to cosmological *N*-body simulations. Astrophys. J. Suppl. Ser. **75**, 231 (1991). doi:10.1086/191530
Hockney, RW, Eastwood, JW: Computer Simulation Using Particles. Hilger, Bristol (1988).
Holmberg, E: On the clustering tendencies among the nebulae. II. A study of encounters between laboratory models of stellar systems by a new integration procedure. Astrophys. J. **94**, 385 (1941). doi:10.1086/144344
Ishiyama, T, Fukushige, T, Makino, J: GreeM : massively parallel TreePM code for large cosmological *N*-body simulations. Publ. Astron. Soc. Jpn. **61**(6), 1319-1330 (2009). doi:10.1093/pasj/61.6.1319; arXiv:0910.0121
Ishiyama, T, Nitadori, K, Makino, J: 4.45 Pflops astrophysical *N*-body simulation on K computer - the gravitational trillion-body problem. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC (2012). doi:10.1109/SC.2012.3; arXiv:1211.4406
Klessen, R: GRAPESPH with fully periodic boundaries: fragmentation of molecular clouds. In: Clarke, DA, West, MJ (eds.) Computational Astrophysics: 12th Kingston Meeting on Theoretical Astrophysics. Astronomical Society of the Pacific Conference Series, vol. 123, p. 169 (1997)
Laureijs, R, et al.: Euclid definition study report (2011). arXiv:1110.3193
LSST Science Collaboration, et al.: LSST Science Book, Version 2.0 (2009). arXiv:0912.0201
Peebles, PJE: Structure of the coma cluster of galaxies. Astron. J. **75**, 13 (1970). doi:10.1086/110933
Reed, DS, Smith, RE, Potter, D, Schneider, A, Stadel, J, Moore, B: Toward an accurate mass function for precision cosmology. Mon. Not. R. Astron. Soc. **431**(2), 1866-1882 (2012). doi:10.1093/mnras/stt301; arXiv:1206.5302
Schneider, A, Teyssier, R, Potter, D, Stadel, J, Onions, J, Reed, DS, Smith, RE, Springel, V, Pearce, FR, Scoccimarro, R: Matter power spectrum and the challenge of percent accuracy. J. Cosmol. Astropart. Phys. **2016**(4), 047 (2016). doi:10.1088/1475-7516/2016/04/047; arXiv:1503.05920

Skillman, SW, Warren, MS, Turk, MJ, Wechsler, RH, Holz, DE, Sutter, PM: Dark sky simulations: early data release (2014). arXiv:1407.2600
Spergel, DN, Verde, L, Peiris, HV, Komatsu, E, Nolta, MR, Bennett, CL, Halpern, M, Hinshaw, G, Jarosik, N, Kogut, A, Limon, M, Meyer, SS, Page, L, Tucker, GS, Weiland, JL, Wollack, E, Wright, EL: First year Wilkinson microwave anisotropy probe (WMAP) observations: determination of cosmological parameters. Astrophys. J. Suppl. Ser. **148** 175-194 (2003). doi:10.1086/377226; arXiv:astro-ph/0302209
Spergel, D, et al.: Wide-field infrared survey telescope-astrophysics focused telescope assets WFIRST-AFTA final report (2013). arXiv:1305.5422
Springel, V: The cosmological simulation code GADGET-2. Mon. Not. R. Astron. Soc. **364**(4), 1105-1134 (2005). doi:10.1111/j.1365-2966.2005.09655.x; arXiv:astro-ph/0505010
Stadel, JG: Cosmological *N*-body simulations and their analysis. PhD thesis, University of Washington (2001)
Teyssier, R: Cosmological hydrodynamics with adaptive mesh refinement: a new high resolution code called RAMSES. Astron. Astrophys. **385**(1), 337-364 (2001). doi:10.1051/0004-6361:20011817; arXiv:astro-ph/0111367
Warren, MS, Salmon, JK: A parallel hashed oct-tree *N*-body algorithm. In: Proceedings of the 1993 ACM/IEEE Conference on Supercomputing - Supercomputing '93, pp. 12-21. ACM, New York (1993). doi:10.1145/169627.169640
Warren, MS: 2HOT. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13, pp. 1-12. ACM, New York (2013). doi:10.1145/2503210.2503220; arXiv:1310.4502
Warren, MS, Goda, MP, Becker, DJ, Salmon, JK, Winckelmans, GS, Sterling, T: Pentium Pro inside: I. A treecode at 430 Gigaflops on ASCI Red, II. Price/performance of $50/Mflop on Loki and Hyglac. In: Proceedings of Supercomputing '97 (1997)